

An algorithmic approach using multivariate polynomials for the nonlinearity of Boolean functions

E. Bellini ^{*} T. Mora [†] M. Sala [‡]

Abstract

The nonlinearity of a Boolean function is a key property in deciding its suitability for cryptographic purposes, e.g. as a combining function in stream ciphers, and so the nonlinearity computation is an important problem for applications. Traditional methods to compute the nonlinearity are based on transforms, such as the Fast Walsh Transform. In 2007 Simonetti proposed a method to solve the above problem seen as a decision problem on the existence of solutions for some multivariate polynomial systems. Although novel as approach, her algorithm suffered from a direct application of Gröbner bases and was thus impractical. We now propose two more practical approaches, one that determines the existence of solutions for Simonetti's systems in a faster way and another that writes similar systems but over fields with a different characteristics. For our algorithms we provide an efficient implementation in the software package MAGMA.

Keywords: Boolean function, Cryptography, multivariate polynomials, Gröbner bases

Introduction

Any function from $(\mathbb{F}_2)^n$ to \mathbb{F}_2 is called a Boolean function. Boolean functions are important in symmetric cryptography, since they are used in the confusion layer of ciphers. An affine Boolean function does not provide an effective confusion. To overcome this, we need functions which are as far as possible from being an affine function. The effectiveness of these functions is measured by several parameters, one of these is called nonlinearity ([Car10]). Traditional methods to compute the nonlinearity are based on transforms, such as the Fast Walsh Transform. In [SS07] a method was proposed by Simonetti that was based on an interpretation of the above problem as a

^{*}eemanuele.bellini@gmail.com

[†]theomora@disi.unige.it

[‡]maxsalacodes@gmail.com

decision problem on the existence of solutions for some multivariate polynomial systems with Boolean variables. This was the first time that a classical problem in Boolean functions was tackled with multivariate polynomial techniques. Although novel and interesting as approach, her algorithm suffered from a direct application of Gröbner bases and was thus impractical. In this paper we propose two more practical approaches, one that determines the existence of solutions for Simonetti’s systems in an efficient way and another that writes similar systems but over fields with a different characteristics. For our algorithms we provide an efficient implementation in the software package MAGMA, which was presented at the conference MEGA 2015 [BMS15]. Although the complexity of our method is still far from the best-known methods using Fast Walsh Transforms or similar techniques, we believe that our improvement on Simonetti’s original idea is significant and that there is still space for improvement on multivariate-polynomials methods to solve this problem, which might also lead to new insights.

The structure of this paper is as follows. In Sections 1 we recall our preliminaries, especially regarding Boolean functions and related polynomials. In Section 2 we describe Simonetti’s approach. In Section 3 we sketch a strategy to solve Simonetti’s systems which does not require the computation of a Groebner basis, thanks to a 1992 idea by Traverso, and we thus describe our first algorithm, Algorithm 1, which is a refined version of Simonetti’s algorithm. In Section 4 we associate to each Boolean function in n variables a rational polynomial whose evaluations represent the distance from all possible affine functions, obtaining our second algorithm, Algorithm 2. In Section 5, we use an alternative approach to compute the nonlinearity avoiding the hard task of solving a polynomial system of equations, which is described in Algorithm 3.

In Section 6 we provide some complexity considerations.

1 Preliminaries

1.1 Nonlinearity of Boolean functions

In this section we summarize some definitions and known results from [Car10] and [MS77], concerning Boolean functions and the classical techniques to determine their nonlinearity.

We denote by \mathbb{F} the field \mathbb{F}_2 . The set \mathbb{F}^n is the set of all binary vectors of length n , viewed as an \mathbb{F} -vector space.

Let $v \in \mathbb{F}^n$. The *Hamming weight* $w(v)$ of the vector v is the number of its nonzero coordinates. For any two vectors $v_1, v_2 \in \mathbb{F}^n$, the *Hamming distance* between v_1 and v_2 , denoted by $d(v_1, v_2)$, is the number of coordinates in which the two vectors differ.

A *Boolean function* is a function $f : \mathbb{F}^n \rightarrow \mathbb{F}$. The set of all Boolean functions

from \mathbb{F}^n to \mathbb{F} will be denoted by \mathcal{B}_n .

We assume implicitly to have ordered \mathbb{F}^n , so that $\mathbb{F}^n = \{\mathbf{p}_1, \dots, \mathbf{p}_{2^n}\}$.

A Boolean function f can be specified by a *truth table*, which gives the evaluation of f at all \mathbf{p}_i 's.

Definition 1.1. *We consider the evaluation map:*

$$\mathcal{B}_n \longrightarrow \mathbb{F}^{2^n} \quad f \longmapsto \underline{f} = (f(\mathbf{p}_1), \dots, f(\mathbf{p}_{2^n})).$$

The vector \underline{f} is called the evaluation vector of f .

Once the order on \mathbb{F}^n is chosen, i.e. the \mathbf{p}_i 's are fixed, it is clear that the evaluation vector of f uniquely identifies f .

A Boolean function $f \in \mathcal{B}_n$ can be expressed in a unique way as a square free polynomial in $\mathbb{F}[X] = \mathbb{F}[x_1, \dots, x_n]$, i.e.

$$f = \sum_{v \in \mathbb{F}^n} b_v X^v,$$

where $X^v = x^{v_1} \dots x^{v_n}$.

This representation is called the *Algebraic Normal Form* (ANF).

Definition 1.2. *The degree of the ANF of a Boolean function f is called the algebraic degree of f , denoted by $\deg f$, and it is equal to $\max\{w(v) \mid v \in \mathbb{F}^n, b_v \neq 0\}$.*

Let \mathcal{A}_n be the set of all affine functions from \mathbb{F}^n to \mathbb{F} , i.e. the set of all Boolean functions in \mathcal{B}_n with algebraic degree 0 or 1. If $\alpha \in \mathcal{A}_n$ then its ANF can be written as

$$\alpha(X) = a_0 + \sum_{i=1}^n a_i x_i.$$

There exists a well known divide-and-conquer butterfly algorithm (see [Car10], p.10) to compute the ANF from the truth-table (or vice-versa) of a Boolean function, which requires $\mathcal{O}(n2^n)$ bit sums, while $\mathcal{O}(2^n)$ bits must be stored. This algorithm is known as the *fast Möbius transform*.

In [CG99] a useful representation of Boolean functions for characterizing several cryptographic criteria (see also [CG01], [Car02]) is introduced.

Boolean functions can be represented as elements of $\mathbb{K}[X]/\langle X^2 - X \rangle$, where $\langle X^2 - X \rangle$ is the ideal generated by the polynomials $x_1^2 - x_1, \dots, x_n^2 - x_n$, and \mathbb{K} is \mathbb{Z} , \mathbb{Q} , \mathbb{R} , or \mathbb{C} .

Definition 1.3. *Let f be a function on \mathbb{F}^n taking values in a field \mathbb{K} . We call the numerical normal form (NNF) of f the following expression of f as a polynomial:*

$$f(x_1, \dots, x_n) = \sum_{u \in \mathbb{F}^n} \lambda_u \left(\prod_{i=1}^n x_i^{u_i} \right) = \sum_{u \in \mathbb{F}^n} \lambda_u X^u,$$

with $\lambda_u \in \mathbb{K}$ and $u = (u_1, \dots, u_n)$.

It can be proved that any Boolean function f admits a unique numerical normal form. As for the ANF, it is possible to compute the NNF of a Boolean function from its truth table by means of an algorithm similar to a fast Fourier transform, thus requiring $\mathcal{O}(n2^n)$ additions over \mathbb{K} and storing $\mathcal{O}(2^n)$ elements of \mathbb{K} .

From now on let $\mathbb{K} = \mathbb{Q}$.

The truth table of f can be recovered from its NNF by the formula

$$f(u) = \sum_{a \preceq u} \lambda_a, \forall u \in \mathbb{F}^n,$$

where $a \preceq u \iff \forall i \in \{1, \dots, n\} a_i \leq u_i$. Conversely, it is possible to derive an explicit formula for the coefficients of the NNF by means of the truth table of f .

Proposition 1.4. *Let f be any integer-valued function on \mathbb{F}^n . For every $u \in \mathbb{F}^n$, the coefficient λ_u of the monomial X^u in the NNF of f is:*

$$\lambda_u = (-1)^{w(u)} \sum_{a \in \mathbb{F}^n | a \preceq u} (-1)^{w(a)} f(a). \quad (1)$$

Definition 1.5. *Let $f, g \in \mathcal{B}_n$. The distance $d(f, g)$ between f and g is the number of $v \in \mathbb{F}^n$ such that $f(v) \neq g(v)$.*

The following lemma is obvious:

Lemma 1.6. *Let f, g be two Boolean functions. Then*

$$d(f, g) = d(\underline{f}, \underline{g}) = w(\underline{f} + \underline{g}).$$

Definition 1.7. *Let $f \in \mathcal{B}_n$. The nonlinearity of f is the minimum of the distances between f and any affine function*

$$N(f) = \min_{\alpha \in \mathcal{A}_n} d(f, \alpha).$$

The maximum nonlinearity for a Boolean function f is bounded by:

$$\max\{N(f) \mid f \in \mathcal{B}_n\} \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (2)$$

Definition 1.8. *The Walsh transform of a Boolean function $f \in \mathcal{B}_n$ is the following function:*

$$\hat{F} : \mathbb{F}^n \longrightarrow \mathbb{Z} \quad x \longmapsto \sum_{y \in \mathbb{F}^n} (-1)^{x \cdot y + f(y)}.$$

where $x \cdot y$ is the scalar product of x and y .

We have the following fact:

Fact 1.9.

$$N(f) = \min_{v \in \mathbb{F}^n} \{2^{n-1} - \frac{1}{2} \hat{F}(v)\} = 2^{n-1} - \frac{1}{2} \max_{v \in \mathbb{F}^n} \{\hat{F}(v)\}$$

Definition 1.10. *The set of integers $\{\hat{F}(v) \mid v \in \mathbb{F}^n\}$ is called the Walsh spectrum of the Boolean function f .*

It is possible to compute the Walsh spectrum of f from its evaluation vector in $\mathcal{O}(n2^n)$ integer operations, while storing $\mathcal{O}(2^n)$ integers, by means of the *fast Walsh transform* (the Walsh transform is the Fourier transform of the sign function of f). Thus the computation of the nonlinearity of a Boolean function f , when this is given either in its ANF or in its evaluation vector, requires $\mathcal{O}(n2^n)$ integer operations and a memory of $\mathcal{O}(2^n)$.

Faster methods are known in particular cases, for example when the ANF is a sparse polynomial [cc13a], [cc13b].

1.2 Polynomials and vector weights

Here we present some results from [SS07], [Sim09], [GOS10]. Let \mathbb{K} be a field and $X = \{x_1, \dots, x_s\}$ be a set of variables. We denote by $\mathbb{K}[X]$ the multivariate polynomial ring in the variables X . If $f_1, \dots, f_N \in \mathbb{K}[X]$, we denote by $\langle \{f_1, \dots, f_N\} \rangle$ the ideal in $\mathbb{K}[X]$ generated by f_1, \dots, f_N . Let I be an ideal in $\mathbb{K}[X]$, we denote by $\mathcal{V}(I)$ its variety, that is the set of its zeros in the algebraic closure of \mathbb{K} .

Let q be the power of a prime. We denote by $E_q[X] = \{x_1^q - x_1, \dots, x_s^q - x_s\}$, the set of field equations in $\mathbb{F}_q[X] = \mathbb{F}_q[x_1, \dots, x_s]$, where $s \geq 1$ is an integer, understood from now on. We write $E[X]$ when $q = 2$.

Definition 1.11. *Let $1 \leq t \leq s$ and $\mathbf{m} \in \mathbb{F}_q[X]$. We say that \mathbf{m} is a **square free monomial of degree t** (or a **simple t -monomial**) if:*

$$\mathbf{m} = x_{h_1} \cdots x_{h_t}, \text{ where } h_1, \dots, h_t \in \{1, \dots, s\} \text{ and } h_\ell \neq h_j, \forall \ell \neq j,$$

i.e. a monomial in $\mathbb{F}_q[X]$ such that $\deg_{x_{h_i}}(\mathbf{m}) = 1$ for any $1 \leq i \leq t$, and 0 otherwise. We denote by $\mathcal{M}_{s,t}$ the set of all square free monomials of degree t in $\mathbb{F}_q[X]$.

Let $t \in \mathbb{N}$, with $1 \leq t \leq s$ and let $I_{s,t} \subset \mathbb{F}_q[X]$ be the following ideal

$$I_{s,t} = \langle \{\sigma_t, \dots, \sigma_s\} \cup E_q[X] \rangle,$$

where σ_i are the elementary symmetric functions:

$$\begin{aligned} \sigma_1 &= x_1 + x_2 + \cdots + x_s, \\ \sigma_2 &= x_1x_2 + x_1x_3 + \cdots + x_1x_s + x_2x_3 + \cdots + x_{s-1}x_s, \\ &\dots \\ \sigma_{s-1} &= x_1x_2x_3 \cdots x_{s-2}x_{s-1} + \cdots + x_2x_3 \cdots x_{s-1}x_s, \\ \sigma_s &= x_1x_2 \cdots x_{s-1}x_s. \end{aligned}$$

We also denote by $I_{s,s+1}$ the ideal $\langle E_q[X] \rangle$. For any $1 \leq i \leq s$, let P_i be the set which contains all vectors in $(\mathbb{F}_q)^n$ of weight i , $P_i = \{v \in \mathbb{F}_q^n \mid w(v) = i\}$, and let Q_i be the set which contains all vectors of weight up to i , $Q_i = \sqcup_{0 \leq j \leq i} P_j$.

Theorem 1.12. *Let t be an integer such that $1 \leq t \leq s$. Then the vanishing ideal $\mathcal{I}(Q_t)$ of Q_t is*

$$\mathcal{I}(Q_t) = I_{s,t+1},$$

and its reduced Gröbner basis G is

$$\begin{aligned} G &= E_q[X] \cup \mathcal{M}_{s,t}, & \text{for } t \geq 2, \\ G &= \{x_1, \dots, x_s\}, & \text{for } t = 1. \end{aligned}$$

Let $\mathbb{F}_q[Z]$ be a polynomial ring over \mathbb{F}_q . Let $\mathbf{m} \in \mathcal{M}_{s,t}$, $\mathbf{m} = z_{h_1} \cdots z_{h_t}$. For any polynomial vector W in the module $(\mathbb{F}_q[Z])^n$, $W = (W_1, \dots, W_n)$, we denote by $\mathbf{m}(W)$ the following polynomial in $\mathbb{F}_q[Z]$:

$$\mathbf{m}(W) = W_{h_1} \cdot \dots \cdot W_{h_t}.$$

1.3 A method for the quotient algebra of zero-dimensional ideals

We briefly recall the notions of Gröbner description, natural representation (or Gröbner representation) and linear representation, which can be found in [Mor05].

Let $X = x_1, \dots, x_k$. Let \mathcal{M} be the set of monomials in $\mathbb{K}[X]$. Let $J \subset \mathbb{K}[X]$ be a zero-dimensional ideal, $\deg(J) = s$, and denote $\mathbf{A} := \mathbb{K}[X]/J$ the corresponding quotient algebra, which satisfies $\dim_{\mathbb{K}}(\mathbf{A}) = s$.

For any $f \in \mathbb{K}[X]$, we will denote $[f] \in \mathbf{A}$ its residue class modulo J and Φ_f the endomorphism $\Phi_f : \mathbf{A} \rightarrow \mathbf{A}$ defined by

$$\Phi_f([g]) = [fg] \forall [g] \in \mathbf{A}.$$

If we fix any \mathbb{K} -basis $\mathbf{b} = \{[b_1], \dots, [b_s]\}$ of \mathbf{A} so that $\mathbf{A} = \text{span}_{\mathbb{K}}(\mathbf{b})$, then for each $g \in \mathbb{K}[X]$, there is a unique (row) vector, the *Gröbner description* of g ,

$$\mathbf{Rep}(g, \mathbf{b}) := (\gamma(g, b_1, \mathbf{b}), \dots, \gamma(g, b_s, \mathbf{b})) \in \mathbb{K}^s$$

which satisfies

$$[g] = \sum_j \gamma(g, b_j, \mathbf{b}) [b_j]$$

and the endomorphism Φ_f is naturally represented by the square matrix

$$M([f], \mathbf{b}) = (\gamma(fb_i, b_j, \mathbf{b})) : \Phi_f(b_i) = [fb_i] = \sum_j \gamma(fb_i, b_j, \mathbf{b}) [b_j].$$

Definition 1.13. A natural representation of J is the assignment of

- a \mathbb{K} -basis $\mathbf{b} = \{[b_1], \dots, [b_s]\} \subset \mathbf{A}$ and
- the square matrices $A_h := \left(a_{ij}^{(h)}\right) = M([x_h], \mathbf{b})$ for each $h, 1 \leq h \leq k$.

Remark that, for each $f(x_1, \dots, x_k) \in \mathbb{K}[X]$, $M([f], \mathbf{b}) = f(A_1, \dots, A_k)$. An equivalent (via the remark above) definition of natural representation can require the further assignment of

- s^3 values $\gamma_{ij}^{(l)} \in \mathbb{K}$ such that

$$[b_i b_j] = \sum_l \gamma_{ij}^{(l)} [b_l]$$

for each $i, j, l, 1 \leq i, j, l \leq s$.

This notion was introduced in [Tra92b, Tra92a] and reconsidered in [AMM03], [Mor05, Definition 29.3.3] under the name of *Gröbner representation*.

The endomorphism Φ_f and its representation $M([f], \mathbf{b})$ were introduced, with f a linear form, in [AS88] as a tool for efficient solving 0-dimensional ideals. If J is given by its Gröbner basis wrt a term-ordering $<$ its natural (actually: “linear” with the definition below) representation can be obtained via [FGLM93, Procedure 3.1].

If J is an affine complete intersection defined by r polynomials a natural representation of it can be efficiently computed via Cardinal-Mourren Algorithm [J.P93, Mou05]. Recalling that a set $N \subset \mathcal{M}$ is called an *escalier* if it is an *order ideal*, i.e. if for each $\lambda, \tau \in \mathcal{M}$, $\lambda\tau \in N \implies \tau \in N$ and properly extending [Mor05, Definition 29.3.3] we set

Definition 1.14. A natural representation is called a linear representation iff the basis \mathbf{b} of the representation is an escalier.

Traverso introduced an algorithm in a scenario related to Gröbner bases computation of a zero-dimensional ideal I (informal talk at MEGA 1992). The setting was reformulated in [Mor05], Algorithm 29.3.8, as follows: given a zero-dimensional ideal $I \subset \mathbb{F}_q[X]$ via its natural representation

$$\mathbf{b} = \{b_1, \dots, b_s\}, b_1 = 1, M := \left\{ \left(a_{lj}^{(h)}\right), 1 \leq h \leq r \right\},$$

and a finite set of elements $F := \{g_1, \dots, g_t\} \subset \mathbb{F}_q[X]$, given via their Gröbner descriptions

$$\mathbf{c}^{(i)} = (c_1^{(i)}, \dots, c_s^{(i)}), c_j^{(i)} = \gamma(g_i, b_j, \mathbf{b}) \forall i, j, 1 \leq i \leq t, 1 \leq j \leq s,$$

so that $g_i - \sum_{j=1}^s c_j^{(i)} b_j \in I$, for each i , compute with good complexity the linear representation of the ideal $J := I \cup \mathbb{I}(F)$.

The basic idea of the algorithm is the following: if we consider an element $g \in F$, having the Gröbner description

$$g - \sum_{j=1}^{\iota} c_j b_j \in \mathfrak{l}, \quad c_{\iota} \neq 0,$$

and we enlarge I by adding g to it, then we obtain the relation

$$b_{\iota} \equiv - \sum_{j=1}^{\iota-1} c_{\iota}^{-1} c_j b_j \pmod{\mathfrak{l} \cup \{g\}};$$

the decomposition $\mathbb{F}_q[X] = I \oplus \text{span}_{\mathbb{F}_q}(\mathbf{b})$ of $\mathbb{F}_q[X]$ into disjoint \mathbb{F}_q -vectorspaces is then transformed into

$$\mathbb{F}_q[X] = (I \cup \{g\}) \oplus \text{span}_{\mathbb{F}_q}(\mathbf{b} \setminus \{b_{\iota}\}),$$

and we only have to substitute, in each Gröbner description $\sum_{j=1}^s d_j b_j$ of the polynomials g_i and $X_h b_l$ — which are respectively encoded in the vectors $\mathbf{c}^{(i)}$ and in the rows $(a_{l1}^{(h)}, \dots, a_{ls}^{(h)})$ of the matrices of M — the instances of b_{ι} with $-\sum_{j=1}^{\iota-1} c_{\iota}^{-1} c_j b_j$ thus getting $\sum_j (d_j - c_{\iota}^{-1} c_j d_{\iota}) b_j$.

Since J is an ideal, the inclusion in it of g implies that J necessarily contains also the polynomials $X_h g$; note that, if the current natural representation is

$$(\mathbf{b}', M') : \mathbf{b}' := \{b'_1, \dots, b'_{\sigma}\}, M' = M(\mathbf{b}') := \left\{ \left(d_{lj}^{(h)} \right) \right\}$$

and $g = \sum_{l=1}^s c_l b'_l$ then

$$X_h g = \sum_{l=1}^s c_l X_h b'_l = \sum_{j=1}^s \left(\sum_{l=1}^s c_l d_{lj}^{(h)} \right) b_j$$

which must be inserted in the list F in order to be treated in the same way.

At termination, if $H \subset \{1, \dots, n\}$ denotes the set of indices of the elements b_j which have not being removed from \mathbf{b} in this procedure, then J is described by the natural representation

$$\mathbf{b}' = \{b_j, i \in H\}, M' = \left\{ \left(a_{lj}^{(h)} \right), l, j \in H, 1 \leq h \leq n \right\}.$$

We observe that Traverso's Algorithm needs to perform at most s **While**-loops, each costing $\mathcal{O}(ns^2)$

2 Simonetti's polynomial systems for the nonlinearity

In this section we want to tackle the following problem: to find a method to compute the nonlinearity of a given Boolean function $f \in \mathcal{B}_n$ by constructing a finite number of polynomial systems over \mathbb{F}_2 with N variables and such that:

- A) N is of the order of n ,
- B) the nonlinearity is obtained by merely deciding which of these systems have a binary solution.

Since the maximum nonlinearity is of the order of 2^{n-1} , we are satisfied if the number of systems we have to construct does not exceed 2^{n-1} .

In this section we report the solution of the above problem, given by Simonetti in [SS07], which depends on Theorem 1.12. The starting idea is to define an ideal such that a point in its variety corresponds to an affine function with distance at most $t - 1$ from f .

Let A be the variable set $A = \{a_i\}_{0 \leq i \leq n}$. We denote by $\mathbf{g}_n \in \mathbb{F}[A, X]$ the following polynomial:

$$\mathbf{g}_n = a_0 + \sum_{i=1}^n a_i x_i .$$

According to Lemma 1.6, determining the nonlinearity of $f \in \mathcal{B}_n$ is the same as finding the minimum weight of the vectors in the set $\{\underline{f} + \underline{g} \mid g \in \mathcal{A}_n\} \subset \mathbb{F}^{2^n}$. We can consider the evaluation vector of the polynomial \mathbf{g}_n as follows:

$$\underline{\mathbf{g}}_n = (\mathbf{g}_n(A, \mathbf{p}_1), \dots, \mathbf{g}_n(A, \mathbf{p}_{2^n})) \in (\mathbb{F}[A])^{2^n} .$$

Definition 2.1. We denote by $J_t^n(f)$ the ideal in $\mathbb{F}[A]$:

$$\begin{aligned} J_t^n(f) &= \langle \{\mathbf{m}(\mathbf{g}_n(A, \mathbf{p}_1) + f(\mathbf{p}_1), \dots, \mathbf{g}_n(A, \mathbf{p}_{2^n}) + f(\mathbf{p}_{2^n})) \mid \mathbf{m} \in \mathcal{M}_{2^n, t}\} \cup E[A] \rangle \\ &= \langle \{\mathbf{m}(\underline{\mathbf{g}}_n + \underline{f}) \mid \mathbf{m} \in \mathcal{M}_{2^n, t}\} \cup E[A] \rangle . \end{aligned}$$

Remark 2.2. As $E[A] \subset J_t^n(f)$, $J_t^n(f)$ is zero-dimensional and radical.

Lemma 2.3. For $1 \leq t \leq 2^n$ the following statements are equivalent:

1. $\mathcal{V}(J_t^n(f)) \neq \emptyset$,
2. $\exists u \in \{\underline{f} + \underline{g} \mid g \in \mathcal{A}_n\}$ such that $w(u) \leq t - 1$,
3. $\exists \alpha \in \mathcal{A}_n$ such that $d(f, \alpha) \leq t - 1$.

From Lemma 2.3 we immediately have the following theorem.

Theorem 2.4. *Let $f \in \mathcal{B}_n$. The nonlinearity $N(f)$ is the minimum t such that $\mathcal{V}(J_{t+1}^n(f)) \neq \emptyset$.*

From this theorem we can derive an algorithm to compute the nonlinearity for a function $f \in \mathcal{B}_n$, by determining if the variety of the ideal $J_t^n(f)$ has a solution or not.

Algorithm 1 Basic algorithm to compute the nonlinearity of a Boolean function by finding if a solution of a polynomial systems over \mathbb{F} exists

Input: a Boolean function f

Output: the nonlinearity of f

```

1:  $j \leftarrow 1$ 
2: while  $\mathcal{V}(J_j^n(f)) = \emptyset$  do
3:    $j \leftarrow j + 1$ 
4: end while
5: return  $j - 1$ 

```

Simonetti's systems $J_j^n(f)$ are the solutions of the problem we stated at the beginning of this section: they use only $n + 1$ variables and all we want to know from them (in the worst case) is whether they have a solution or not. Observe also that the solution we are interested in does not lie in some extension field but it must remain in $(\mathbb{F}_2)^{n+1}$.

Moreover, the number of systems we need to check is, in the worst case, the maximum nonlinearity plus one. We claim that with our constraints Simonetti's solution is, in principle, still the best-known.

However, a practical application of Algorithm 1 was missing in Simonetti's work, where she would use straightforward applications of Gröbner bases.

Remark 2.5. If f is not affine, we can start our check from $J_2^n(f)$.

3 A faster algorithm for solving Simonetti's systems

As we have seen in Section 2, the nonlinearity of a Boolean function can be computed solving polynomial systems over \mathbb{F} . It is sufficient to find the minimum j such that the variety of the ideal $J_t^n(f)$ is not empty. Recall that

$$J_t^n(f) = \langle \{ \mathbf{m}(\underline{\mathbf{g}}_n + \underline{f}) \mid \mathbf{m} \in \mathcal{M}_{2^n, t} \} \cup E[A] \rangle.$$

This method becomes impractical even for small values of n , since $\binom{2^n}{t}$ monomials have to be evaluated. A first slight improvement could be achieved by adding to the ideal one monomial evaluation at a time and check if 1 has appeared in the Gröbner basis. Even this way, the algorithm remains very slow.

Of course, an actual implementation would take care to reduce modulo the temporary basis any monomial before adding it to the computation, but it would still remain too slow. To achieve a real improvement we need to use Traverso's strategy explained in Section 1.3. In particular, our proposal is to start with a trivial monomial basis given by the all monomials in $N(E(X))$ and then adding a monomial $\mathbf{m}(\underline{\mathbf{g}}_n + \underline{f})$ at a time, computing the new Hilbert staircase and the associated algebra-multiplication matrix. This way we will not get at the end a Gröbner basis, but we would see from the final Hilbert staircase whether the ideal is trivial or not. Since only linear algebra operations are required, this method is much faster and probably is the fastest that can be used to solve Simonetti's systems. However, in the next section we will propose an even better method, by modifying Simonetti's systems to other fields.

Theorem 3.1. *Solving the reduced Simonetti's systems $J_j^n(f)$ using Traverso's algorithm requires $\mathcal{O}(n2^{2n})$ elementary operations.*

Proof. As noted at the end of Section 1.3, Traverso's algorithm requires $\mathcal{O}(rs^2)$ elementary operations, where r is the number of variables of the equations in the system and s is the number of monomials in the escalier. In our case $r = n + 1$ and $s \leq (2^{n+1})^2$. \square

Actually the complexity exposed in the previous theorem is a large upper bound of the real complexity, since the monomials in the escalier could be much less than $(2^{n+1})^2$, though a precise estimation of their exact number is not known to the authors.

Since the nonlinearity of a Boolean function is bounded by $2^{n-1} - 2^{n/2-1}$, Algorithm 1 implies we have to solve at most 2^{n-1} Simonetti's systems.

On the other hand only the last system dominates the computation since the previous ones do not have a solution.

If we suppose such systems were somehow given for free, computing the nonlinearity of the Boolean function with Algorithm 1 would require $\mathcal{O}(n2^{3n})$ operations.

4 Nonlinearity and polynomial systems over \mathbb{Q}

Here we present an algorithm to compute the nonlinearity of a Boolean function by solving a polynomial system of equations over \mathbb{Q} rather than over \mathbb{F} , which turns out to be much faster than Algorithm 1. The same algorithm can be slightly modified to work over the field \mathbb{F}_p , where p is a prime. The complexity of these algorithms will be analyzed in Section 6.

For each $i = 1, \dots, 2^n$, let us denote:

$$f_i^{(\mathbb{F})}(A) = \mathbf{g}_n(A, \mathbf{p}_i) + f(\mathbf{p}_i)$$

the Boolean function where as usual $A = \{a_0, \dots, a_n\}$ are the $n+1$ variables representing the coefficient of a generic affine function.

In this case we have that:

$$(f_1^{(\mathbb{F})}(A), \dots, f_{2^n}^{(\mathbb{F})}(A)) = \underline{g}_n(A) + \underline{f} \in (\mathbb{F}[A])^{2^n}$$

Note that the polynomials $f_i^{(\mathbb{F})}$ are affine polynomials.

We also denote by

$$f_i^{(\mathbb{Z})}(A) = \text{NNF}(f_i^{(\mathbb{F})}(A))$$

the NNF of each $f_i^{(\mathbb{F})}(A)$ (obtained as in [CG99], Theorem 1).

Definition 4.1. We call $\mathbf{n}_f(A) = f_1^{(\mathbb{Z})}(A) + \dots + f_{2^n}^{(\mathbb{Z})}(A) \in \mathbb{Z}[A]$ the **integer nonlinearity polynomial** (or simply the nonlinearity polynomial) of the Boolean function f .

For any $t \in \mathbb{N}$ we define the ideal $\mathcal{N}_f^t \subseteq \mathbb{Q}[A]$ as follows:

$$\mathcal{N}_f^t = \langle E[A] \bigcup \{f_1^{(\mathbb{Z})} + \dots + f_{2^n}^{(\mathbb{Z})} - t\} \rangle = \quad (3)$$

$$= \langle E[A] \bigcup \{\mathbf{n}_f - t\} \rangle \quad (4)$$

Note that the evaluation vector $\underline{\mathbf{n}}_f$ represents all the distances of f from all possible affine functions (in n variables).

Theorem 4.2. The variety of the ideal \mathcal{N}_f^t is non-empty if and only if the Boolean function f has distance t from an affine function. In particular, $N(f) = t$, where t is the minimum positive integer such that $\mathcal{V}(\mathcal{N}_f^t) \neq \emptyset$.

Proof. Note that

$$\mathcal{N}_f^t = \langle E[A] \rangle + \langle \{\mathbf{n}_f(A) - t\} \rangle$$

and so

$$\mathcal{V}(\mathcal{N}_f^t) = \mathcal{V}(\langle E[A] \rangle) \cap \mathcal{V}(\langle \{\mathbf{n}_f(A) - t\} \rangle).$$

Therefore $\mathcal{V}(\mathcal{N}_f^t) \neq \emptyset$ if and only if $\exists \bar{a} = (\bar{a}_0, \dots, \bar{a}_n) \in \mathcal{V}(\langle E[A] \rangle)$ such that $\mathbf{n}_f(\bar{a}) = t$.

Let $\alpha \in \mathcal{A}_n$ such that $\alpha(X) = \bar{a}_0 + \sum_{i=1}^n \bar{a}_i x_i$.

By definition we have

$$f_i^{(\mathbb{Z})} = 1 \iff f(\mathbf{p}_i) \neq \alpha(\mathbf{p}_i)$$

and

$$f_i^{(\mathbb{Z})} = 0 \iff f(\mathbf{p}_i) = \alpha(\mathbf{p}_i).$$

Hence

$$\begin{aligned} \mathbf{n}_f(\bar{a}) = \sum_{i=1}^{2^n} f_i^{(\mathbb{Z})}(\bar{a}) - t = 0 &\iff |\{i \mid f(\mathbf{p}_i) \neq \alpha(\mathbf{p}_i)\}| = t \\ &\iff d(f, \alpha) = t. \end{aligned}$$

and our claim follows directly. \square

To compute the nonlinearity of f we can use Algorithm 2 with input f .

Algorithm 2 To compute the nonlinearity of the Boolean function f

Input: f

Output: nonlinearity of f

```

1: Compute  $\mathbf{n}_f$ 
2:  $j \leftarrow 1$ 
3: while  $\mathcal{V}(\mathcal{N}_f^j) = \emptyset$  do
4:    $j \leftarrow j + 1$ 
5: end while
6: return  $j$ 

```

Algorithm 2 can be modified to eliminate the while cycle. Instead of checking if a solution of the system

$$\begin{cases} a_0^2 - a_0 = 0 \\ \dots \\ a_n^2 - a_n = 0 \\ \mathbf{n}_C(a_0, \dots, a_n) - j = 0 \end{cases} \quad (5)$$

exists in the affine algebra $\mathbb{Q}/\langle a_0^2 - a_0, \dots, a_n^2 - a_n \rangle$ for each $j \in \{1, \dots, 2^n\}$, we can add the variable t to the system

$$\begin{cases} a_0^2 - a_0 = 0 \\ \dots \\ a_n^2 - a_n = 0 \\ \mathbf{n}_C(a_0, \dots, a_n) - t = 0 \end{cases} \quad (6)$$

and solve it in $\mathbb{Q}[t]/\langle a_0^2 - a_0, \dots, a_n^2 - a_n \rangle$, with respect to lexicographical monomial ordering, to find as a solution a polynomial $\mathbf{t}(t)$, whose zeros are integers, representing the possible distances of the Boolean function f from the affine functions. We are interested in the smallest solution of $\mathbf{t}(t)$.

We did not investigate further which of the two solutions is best.

5 An improvement using fast polynomial evaluation

Once the nonlinearity polynomial \mathbf{n}_f is defined, we can use another approach to compute the nonlinearity avoiding the hard task of solving a polynomial system of equations.

We have to find the minimum nonnegative integer t in the set of the evaluations of \mathbf{n}_f , that is, in $\{\mathbf{n}_f(\bar{a}) \mid \bar{a} \in \{0, 1\}^{n+1} \subset \mathbb{Z}^{n+1}\}$.

We write explicitly the modified algorithm.

Algorithm 3 To compute the nonlinearity of the Boolean function f

Input: f

Output: nonlinearity of f

```

1: if  $f \in \mathcal{A}_n$  then
2:   return 0
3: else
4:   Compute  $\mathbf{n}_f$ 
5:   Compute  $m = \min\{\mathbf{n}_f(\bar{a}) \mid \bar{a} \in \{0, 1\}^{n+1}\}$ 
6:   return  $m$ 
7: end if

```

Example 5.1. Consider the case $n = 2$, $f(x_1, x_2) = x_1x_2 + 1$. We have that $\underline{f} = (1, 1, 1, 0)$ and $\underline{\mathbf{g}}_n = (a_0, a_0 + a_1, a_0 + a_2, a_0 + a_1 + a_2)$.

Let us compute all $f_i^{(\mathbb{F})} = (\underline{\mathbf{g}}_n + \underline{f})_i$ and $f_i^{(\mathbb{Z})}$, for $i = 1, \dots, 2^2$:

$$\begin{aligned}
f_1^{(\mathbb{F})} &= a_0 + 1 & \rightarrow f_1^{(\mathbb{Z})} &= -a_0 + 1 \\
f_2^{(\mathbb{F})} &= a_0 + a_1 + 1 & \rightarrow f_2^{(\mathbb{Z})} &= 2a_0a_1 - a_0 - a_1 + 1 \\
f_3^{(\mathbb{F})} &= a_0 + a_2 + 1 & \rightarrow f_3^{(\mathbb{Z})} &= 2a_0a_2 - a_0 - a_2 + 1 \\
f_4^{(\mathbb{F})} &= a_0 + a_1 + a_2 & \rightarrow f_4^{(\mathbb{Z})} &= 4a_0a_1a_2 - 2a_0a_1 - 2a_0a_2 \\
& & & + a_0 - 2a_1a_2 + a_1 + a_2
\end{aligned}$$

Then $\mathbf{n}_f = f_1^{(\mathbb{Z})} + f_2^{(\mathbb{Z})} + f_3^{(\mathbb{Z})} + f_4^{(\mathbb{Z})} = 4a_0a_1a_2 - 2a_0 - 2a_1a_2 + 3$ and since

$$\underline{\mathbf{n}}_f = (3, 1, 3, 1, 3, 1, 1, 3)$$

then the nonlinearity of f is 1.

Observe that the vector $\underline{\mathbf{n}}_f$ represents all the distances of f from all possible affine functions in 2 variables, that is, from $0, 1, x_1, x_1 + 1, x_2, x_2 + 1, x_1 + x_2, x_1 + x_2 + 1$.

6 Complexity considerations

First we recall that the complexity of computing the nonlinearity of a Boolean function with n variables, having as input its coefficients vector, is $\mathcal{O}(n2^n)$ using the Fast Möbius and the Fast Walsh Transform.

We now want to analyze the complexity of Algorithm 1, 2, 3.

The complexity of constructing the nonlinearity polynomial is claimed in an unpublished preprint as follows:

Theorem 6.1 ([BSS14]). *There exists an algorithm to compute the nonlinearity polynomial, which requires:*

1. $\mathcal{O}(n2^n)$ integer sums and doublings.
In particular $n2^n$ integer sums and $n2^{n-1}$ integer doublings, i.e. the big \mathcal{O} constant is $c = 3/2$, provided doubling costs as summing.
2. the storage of $\mathcal{O}(2^n)$ integers of size less than or equal to 2^n .

6.1 Some considerations on Algorithm 1

In Algorithm 1, almost all the computations are wasted evaluating all possible simple- t -monomials in 2^n variables, which are $\binom{2^n}{t}$. This number grows enormously even for small values of n and t . We investigated experimentally how many of the $\binom{2^n}{t}$ monomials are actually needed to compute the final Gröbner basis of J_t^n . Our experiment ran over all possible Boolean functions in 3 and 4 variables. The results are reported in Tables 1, 2 and 3. In this tables, for each J_t^n there are four columns. Let G_t^n be the Gröbner basis of J_t^n .

Under the column labeled #C we report the average number of *checked* monomials in 2^n variables before obtaining G_t^n .

Under the column labeled #S we report the average number of monomials which are actually *sufficient* to obtain G_t^n .

Under the columns labeled “m” e “M” we report, respectively, the minimum and the maximum number of sufficient monomials to find G_t^n running through all possible Boolean functions in n variables.

For example, to compute the Gröbner basis of the ideal J_2^3 associated to a Boolean function f whose nonlinearity is 2, we needed to check on average 24 monomials before finding the correct basis. Between the 24 monomials only 9.7 (on average) were sufficient to obtain the same basis, where the number of sufficient monomials never exceeded the range 8 – 11.

NL	J_1^3				J_2^3				J_3^3			
	#S	m	M	#C	#S	m	M	#C	#S	m	M	#C
0	4	4	4	8	0	0	0	0	0	0	0	0
1	4.5	4	5	4.4	8.5	7	10	28	0	0	0	0
2	4.4	4	5	4	9.7	8	11	24	9.3	8	11	56

Table 1: Number of monomials needed to compute the Gröbner basis of the ideal J_t^3 .

NL	J_1^4				J_2^4				J_3^4			
	#S	m	M	#C	#S	m	M	#C	#S	m	M	#C
0	5	5	5	16	0	0	0	0	0	0	0	0
1	5.25	4	6	8	8.75	8	11	120	0	0	0	0
2	4.83	4	6	5.67	9.97	8	12	62.83	14.50	12	18	560
3	4.62	4	6	4.76	9.92	8	12	42.72	15.76	13	19	315.04
4	4.53	4	6	4.42	9.83	8	12	37.49	15.81	13	19	246.19
5	4.46	4	5	4.19	10.11	8	12	34.39	15.89	13	19	215.68
6	4.43	4	5	4.00	9.71	8	11	24.00	17.29	16	19	156.86

Table 2: Number of monomials needed to compute the Gröbner basis of the ideal J_t^4 , $t = 1, 2, 3$.

NL	J_4^4				J_5^4				J_6^4				J_7^4			
	#S	m	M	#C	#S	m	M	#C	#S	m	M	#C	#S	m	M	#C
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	20.18	15	23	1820	0	0	0	0	0	0	0	0	0	0	0	0
4	21.44	16	24	1319.96	23.99	22	29	4368	0	0	0	0	0	0	0	0
5	21.54	19	24	1003.15	26.00	24	28	3851.24	23.50	22	25	8008	0	0	0	0
6	19.57	19	20	671.71	28	28	28	2603.79	28	28	28	7608.79	16	16	16	11441

Table 3: Number of monomials needed to compute the Gröbner basis of the ideal J_t^4 , $t = 4, 5, 6, 7$.

6.2 Algorithm 1 and 2

Since the ideal $J_t^n(f)$ of Definition 2.1 is derived from the evaluation of $\binom{2^n}{t}$ monomials (generating at most the same number of equations), then the complexity of Algorithm 1 is equivalent to the complexity of solving a polynomial system of at most $\binom{2^n}{t}$ equations of degree d (where $1 < d \leq t$) in $n + 1$ variables over the field \mathbb{F} . This method becomes almost impractical for $n = 5$. We recall that $t \leq 2^{n-1} - 2^{\frac{n}{2}-1}$ (see Equation 2).

The complexity of Algorithm 2 is equivalent to the complexity of solving a polynomial system of only $n + 1$ field equations plus one single polynomial \mathbf{n}_f of degree at most $n + 1$ in $n + 1$ variables over the field \mathbb{Q} (or over a prime field \mathbb{F}_p) with coefficients of size less than or equal to 2^n . Solving the system by computing its Gröbner basis over a prime field \mathbb{F}_p with $p \sim 2^n$ is much faster than computing the same base over \mathbb{Q} . It may be investigated if there are better size for the prime p , or even faster specialized algorithms to solve the system.

7 Acknowledgments

These results appear partially in the first author's PHD thesis and so he would like to thank the second author and the third author (his supervisor). We presented our algorithms at a computation presentation in the conference MEGA [BMS15].

References

- [AMM03] M. E. Alonso, M. G. Marinari, and T. Mora, *The big mother of all dualities: Möller algorithm*, Comm. Algebra **31** (2003), no. 2, 783–818.
- [AS88] W. Auzinger and H. J. Stetter, *An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations*, Internat. Schriftenreihe Numer. Math. **86** (1988), 11–30.
- [BMS15] E. Bellini, T. Mora, and M. Sala, *Algorithmic approach using polynomial systems for the nonlinearity of boolean functions*, Talk at MEGA 2015, Trento, <http://mega2015.science.unitn.it/>, 2015.
- [BSS14] E. Bellini, I. Simonetti, and M. Sala, *Nonlinearity of Boolean functions: an algorithmic approach based on multivariate polynomials*, Preprint <http://arxiv.org/abs/1404.2741>, 2014.
- [Car02] C. Carlet, *On the coset weight divisibility and nonlinearity of resilient and correlation-immune functions*, Sequences and their Applications, Springer, 2002, pp. 131–144.
- [Car10] C. Carlet, *Boolean functions for cryptography and error correcting codes*, Boolean Models and Methods in Mathematics, Computer Science, and Engineering (2010), 257–397.
- [cc13a] Çağdaş Çalık, *Computing cryptographic properties of boolean functions from the algebraic normal form representation*, Ph.D. thesis, Middle East Technical University, 2013.
- [cc13b] ———, *Nonlinearity Computation for Sparse Boolean Functions*, arXiv preprint arXiv:1305.0860 (2013).
- [CG99] C. Carlet and P. Guillot, *A new representation of Boolean functions*, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Springer, 1999, pp. 94–103.
- [CG01] C. Carlet and P. Guillot, *Bent, resilient functions and the Numerical Normal Form*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **56** (2001), 87–96.

- [FGLM93] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora, *Efficient computation of zero-dimensional Gröbner bases by change of ordering*, J. Symbolic Comput. **16** (1993), no. 4, 329–344.
- [GOS10] E. Guerrini, M. Orsini, and M. Sala, *Computing the distance distribution of systematic non-linear codes*, Journal of Algebra and Its Applications **9** (2010), no. 2, 241–256.
- [J.P93] Cardinal J.P., *Dualité et algorithmes itératifs pour la résolution de systèmes polynomiaux*, Ph.D. thesis, 1993.
- [Mor05] T. Mora, *Solving polynomial equation systems. II, Macaulay’s paradigm and Gröbner technology*, Encyclopedia of Mathematics and its Applications, vol. 99, Cambridge University Press, 2005.
- [Mou05] B. Mourrain, *Bezoutian and quotient ring structure*, J. Symbolic Comput. **39** (2005), no. 3-4, 397–415.
- [MS77] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes. I*, North-Holland Publishing Co., Amsterdam, 1977, North-Holland Mathematical Library, Vol. 16.
- [Sim09] I. Simonetti, *On the non-linearity of Boolean functions*, Gröbner Bases, Coding, and Cryptography (M. Mora T. Perret L. Sakata S. Sala and C. Traverso, eds.), RISC Book Series, Springer, Heidelberg, 2009, pp. 409–413.
- [SS07] M. Sala and I. Simonetti, *An algebraic description of Boolean functions*, Proc. of WCC 2007 (2007), 343–349.
- [Tra92a] C. Traverso, *Linear Gröbner methods and “natural” representations of algebraic numbers*, July 1992, Draft.
- [Tra92b] ———, *Natural representation of algebraic numbers*, Conference at MEGA-92, 1992.